REMARKS

Claims 1 - 33 are pending in the application. Claims 1 - 33 have been rejected.

Claims 1 - 33 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Applicant's Admitted Prior Arts (the Schopp APA) in view of Tucker, U.S. Patent No. 6,223,204 (Tucker).

The present invention generally relates to a process for acquiring a mutex that is autonomically adaptive, on a per-mutex basis, to the computation resources that have been consumed during previous attempts to acquire the mutex.

More specifically, the present invention, as set forth by independent claim 1, relates to a method for managing a mutex in a data processing system. The method includes maintaining an average acquisition cost value for a mutex; attempting to acquire the mutex by a first thread, and in response to a determination that the mutex has already been acquired by a second thread, determining to enter a spin state or a sleep state on the first thread based on the average acquisition cost value for the mutex.

The present invention, as set forth by independent claim 12, relates to an apparatus for managing a mutex in a data processing system. The apparatus includes means for maintaining an average acquisition cost value for a mutex, means for attempting to acquire the mutex by a first thread, and means for determining to enter a spin state or a sleep state on the first thread based on the average acquisition cost value for the mutex in response to a determination that the mutex has already been acquired by a second thread.

The present invention, as set forth by independent claim 23, relates to a computer program product on a computer readable medium for use in a data processing system for managing a mutex. The computer program product includes means for maintaining an average acquisition cost value for a mutex, means for attempting to acquire the mutex by a first thread, and means for determining to enter a spin state or a sleep state on the first thread based on the average acquisition cost value for the mutex in response to a determination that the mutex has already been acquired by a second thread.

Figure 3 of the Schopp application sets forth a typical implementation of a spin lock mutex. In this implementation, a determination is made at to whether a mutex is free and unlocked (step 304) and if not, then a check is made as to whether a configurable amount of time has been used by the thread by spinning on the mutex (step 306). If not, then the thread performs a busy-wait loop (step 308), i.e., the thread spins in a loop, as it waits for the mutex to become available. If the thread has already been through steps 302 – 308, then the thread continues to perform the spinning operation by completing another busy-wait loop. After spinning for some period of time, the thread then repeats step 302.

If a mutex is free at step 304, then the mutex is locked on behalf of the thread (step 310) and the thread may proceed to access a shared resource (step 312) without the possibility of colliding with another thread and compromising the integrity of the data that is associated with the shared resource. After the thread has performed its operations with respect to the shared resource, then the thread request that the mutex be released, and the mutex is unlocked 9step 314), thereby concluding the process. After the mutex has been unlocked, the mutex can be used by other concurrently executing threads. If a configurable amount of time has already been used by the thread by spinning on the mutex as determined at step 305, then the thread sleeps on the mutex (step 316), e.g., by calling a kernel function that causes the thread to be put into a sleep state. The thread may sleep for a configurable period of time, or the kernel may have the ability to wake the thread when the mutex has been unlocked. After the thread is awakened, the thread again attempts to acquire the mutex. (See Schopp Figure 3 and ¶¶ 0037, 0038.)

Thus, nowhere in the discussion of Figure 3 of Schopp (or anywhere in Schopp that is referenced a Prior Art) is there a disclosure or suggestion of maintaining an average acquisition cost value for a mutex or determining to enter a spin state or a sleep state on a first thread based on the average acquisition cost value for the mutex, as required by claims 1, 12 and 23.

When discussing the APA of Schopp and Tucker, the Examiner states:

> However Tucker discloses the step of maintaining an average acquisition cost value (i.e. LWP identifier) for a mutex (see col. 5, lines 2-45); and determining to enter a spin state or a sleep state on the first thread based on the average acquisition cost value for the mutex (see col. 4, lines 7-22). Therefore, it would have been obvious to a person of an ordinary skill in the art at the time the

> invention was made to have combined the teachings of Tucker within the system of AAPA because it would reduce the excessive period of time waiting for the thread requests (Office Action, Page 3, lines 2 – 9).

Tucker generally relates to allocating resources in multithreading computing environment. Tucker discloses scheduling multiple light weight processes to run on one or more data processors. Tucker defines light weight processes as kernel entities which are scheduled to run entirely within a kernel level memory region. Threads are scheduled at user level memory onto LWPs. Particular LWPs are in turn scheduled onto particular processors. (See e.g., Tucker, Col. 1, lines 47 – 51.)

A mutex protects data in memory and permits only one thread to access the data at a time. Data pertaining to the running status of each of the light weight processes is stored in one or more kernel data structures which are mapped to the user level. When a thread attempts to acquire a mutex held by another thread, then the kernel data structure is checked to determine the status of the light weight process and its associated thread. The thread attempting to acquire the mutex is caused to sleep or spin according to the current running or not running status of the light weight process. If the light weight process holding a mutex is running, then the thread attempting to acquire the mutex will spin. If the light weight process then holding a mutex is stopped, then the thread attempting to acquire the mutex will block or sleep until awakened.

The portion of Tucker to which the Examiner refers discusses a light weight process (LWP) scheduled call process which uses a light weight process identifier (LWP identifier) as well as a state flag and a preempt flag to determine when to set or change the scheduling of light weight processes. The LWO identifier indicates an effected LWP. The state flag allows a computer to perform adaptive mutexes and thread affinity scheduling. The preempt flag provides a means to shield LWPs from preemption for short periods of time. (See e.g., Tucker Col. 5, lines 3 – 22.)

As with the Schopp APA, nowhere in Tucker is there a disclosure or suggestion of maintaining an average acquisition cost value for a mutex or determining to enter a spin state or a sleep state on a first thread based on the average acquisition cost value for the mutex, as required by claims 1, 12 and 23.

Accordingly, the Schopp APA and Tucker, taken alone or in combination, do not teach or suggest a method for managing a mutex in a data processing system which includes *maintaining an average acquisition cost value for a mutex*; attempting to acquire the mutex by a first thread, and *in response to a determination that the mutex has already been acquired by a second thread, determining to enter a spin state or a sleep state on the first thread based on the average acquisition cost value for the mutex*, all as required by claim 1. Accordingly, claim 1 is allowable over the Schopp APA and Tucker. Claims 2 - 11 depend from claim 1 and are allowable for at least this reason.

The Schopp APA and Tucker, taken alone or in combination, do not teach or suggest an apparatus for managing a mutex in a data processing system which includes *means for maintaining an average acquisition cost value for a mutex*, means for attempting to acquire the mutex by a first thread, and *means for determining to enter a spin state or a sleep state on the first thread based on the average acquisition cost value for the mutex in response to a determination that the mutex has already been acquired by a second thread*, all as required by claim 12. Accordingly, claim 12 is allowable over the Schopp APA and Tucker. Claims 13 - 22 depend from claim 12 and are allowable for at least this reason.

The Schopp APA and Tucker, taken alone or in combination, do not teach or suggest a computer program product on a computer readable medium for use in a data processing system for managing a mutex where he computer program product includes *means for maintaining an average acquisition cost value for a mutex*, means for attempting to acquire the mutex by a first thread, and *means for determining to enter a spin state or a sleep state on the first thread based on the average acquisition cost value for the mutex in response to a determination that the mutex has already been acquired by a second thread*, all as required by claim 23. Accordingly, claim 23 is allowable over the Schopp APA and Tucker. Claims 22 - 33 depend from claim 23 and are allowable for at least this reason.

CONCLUSION

In view of the amendments and remarks set forth herein, the application is believed to be in condition for allowance and a notice to that effect is solicited. Nonetheless, should any issues remain that might be subject to resolution through a telephonic interview, the examiner is requested to telephone the undersigned.

I hereby certify that this correspondence is being electronically submitted to the COMMISSIONER FOR PATENTS via EFS on November 14, 2006.

/Stephen A. Terrile/

Attorney for Applicant(s)

Respectfully submitted,

/Stephen A. Terrile/

Stephen A. Terrile
Attorney for Applicant(s)
Reg. No. 32,946